Adaptive Rate-Power Control under BATS Protocol

1st Yuzhou Tong
School of Science and Engineering
The Chinese University of Hong Kong, Shenzhen
Shenzhen, China
118010283@link.cuhk.edu.cn

***Abstract*—In this project, we plan to develop an adaptive rate and power control scheme for delay sensitive communications over wireless fading channels under the frame of BATS protocol, which allows a relaxation on bit error rate (BER) of each transmission. Our objective is to find a rate-power control policy that maximizes the system throughput subject to a long-term average power constraint. And we consider a LINK-PHY layer joint control approach for a transmitter with a finite buffer size. The channel is modeled as a finite state Markov channel and we use a reinforcement learning approach to learn the channel and search the optimal control policy.**

I. Introduction

IEEE 802.11 wireless systems usually contain rate control algorithms which are designed to adapt the transmission rate between several available rates in response to varying channel conditions. The efficiency of the rate control algorithm in selecting optimal data rates for the channel conditions directly impacts on the throughput of the wireless system.

Minstrel [3] is a popular rate control algorithm widely used in wireless drivers such as *Ath9k*. The algorithm contains three parts: the retry chain mechanism, the rate selection and the statistic calculations. The retry chain consists of four rate-count pairs, named r0/c0,r1/c1,r2/c2, and r3/c3. A packet is first transmitted at rate r0 for c0 attempts. If these attempts are not successful, Minstrel transmits the frame at rate r1 for c1 attempts. The process continues until either the packet is successfully transmitted or ultimately discarded after (c0+c1+c2+c3) unsuccessful transmission attempts. During normal transmission the r-values in the retry chain are chosen as follows: r0 is set to the rate that achieves the highest expected throughput, r1 is the rate with the second highest expected throughput, r2 is the rate with the highest probability of success, and finally r3 is set to the lowest available data rate. *Minstrel* relies on having accurate statistics about the success rate of transmissions at each data rate. Of course, it has to attempt to send packets at each data rate in order to have statistics on them. 10 % of the data frames are sent as sampling transmissions, where a random rate not currently in the retry chain is chosen to sample. The the r-values are chosen as follows: r0 is set to whichever is higher out of the sample rate or the rate with the highest expected throughput, and r1 is set to whichever is lower. r2 and r3 remain the rate with the highest probability of success and the lowest available rate respectively. The retry chain is shown in TABLE I.

TABLE I
Retry Chain Table

Rate	Sampling Transmission		Normal Transmission
	Random <Best	Random >Best	
r0	Best rate	Random rate	Best rate
r1	Random rate	Best rate	Second best rate
r2	Best probability	Best probability	Best probability
r3	Bese rate	Bese rate	Bese rate

Minstrel maintains the probability of success and expected throughput for each data rate as an Exponentially Weighted Moving Average (EWMA). This probability is based on the historical success rate of packet transmissions at each data rate. This probability is used to estimate the throughput of each rate and the retry chain is re-evaluated based on this estimate every 100ms. In each 100ms sampling window, the success rate, R_S, is calculated for each data rate based on the historical observations of packet success and failures as in (1), where N_S is the number of packets transmitted successfully at the data rate and N_T is the total number of attempted at the data rate.

$$R_S = N_S/N_T \tag{1}$$

$$P(t+1) = R_S \times (1-\alpha) + P(t) \times \alpha \tag{2}$$

The R_S value is then used to alter the measured value for the probability of success for each dara rate using the EWMA expression 2. The EWMA parameter α is used to deternmine how much wright is given to the R_S value from the new sampling period. The default value of α is 0.25. Finally, *Minstrel* calculates an expected throughput for each data rate, T, as in 3

$$T = P_{success} \times (B/t) \tag{3}$$

The expected throughput T is the number of bytes B transferred in time t scaled by the probability of success $P_{success}$. This result is an expected throughput based on the stations previous observations of the proportion of packets that have been successfully transmitted at the data rate. The uplink and downlink UDP tests over an 802.11g wireless testbed indicates that Minstrel performs better than fixed rate transmission and other rate control algorithms (e.g., *Onoe*, *SampleRate*) [5].

The testing results shed valuable insight into how this rate control algorithm might be improved. With joint rate-power control, an improved algorithm *Minstrel-Piano* [1] is able to significantly decrease the power levels while maintaining the same link performance. The goal of *Piano* power control

algorithm is to support the rate, and hence, the throughput that *Minstrel* achieves, with the lowest power possible. *Piano* uses 3 types of packets: (1) Reference packets are sent at a reference power (typically a high power level). (2) Sample packets are sent at different power levels for exploration. (3) Data packets are sent at a power level equal to sample power level plus a constant. By sending packets at different power levels, *Piano* aims to adjust the transmission power according to the change of throughput. To make such adjustments, similar to *Minstrel*, *Piano* also maintains an EWMA of reference, sample, and data success probabilities.

These two algorithms are based on the assumption that they have no channel state information (CSI) at both transmitter (Tx) and receiver (Rx) sides. Thus, they could only use sampling packets to estimate the throughput of each rate and power setting. And the proportion of sampling packets should be adjusted according to the channel conditions. The α value balances the use of the current and historical sampling measurements for the rate selection. One algorithm [4] is proposed to dynamically adapting α. In this approach, they designs a sampling mechanism to probe the maximum achievable throughput for different α values. In this sampling mechanism, they use a monitor window of *n* rate adaptations for computing the throughput variation for two α values. And this approach improves the performance of *Minstrel* by 19% on average.

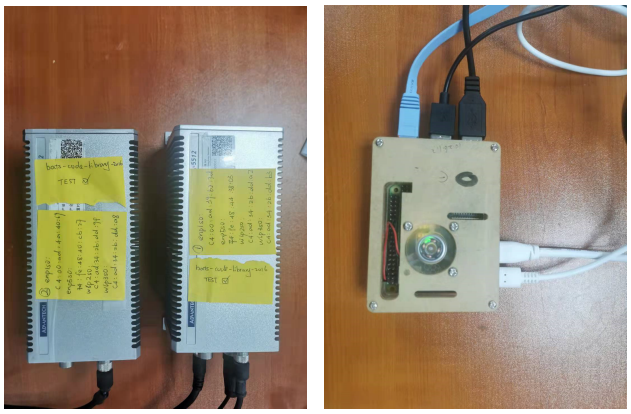
The three algorithms mentioned above rely on sampling mechanism due to the lack of CSI. And there are mainly two limitations for these algorithms: (1) The sampling packets are redundant, and hence, reduce the throughput; (2) There is a delay between the throughput calculation and rate adjustment, especially when you adapt rate, power and other parameters simultaneously. However, if the channel model is appropriately established and channel estimation techniques are used, some advanced resource allocation schemes can be designed to overcome these limitations. One scheme [2] for delay sensitive communications over wireless fading channel was proposed based on reinforcement learning. In this approach, they model the channel fading process as a finite state Markov channel (FSMC) and use channel estimation and feedback approach to get the CSI at both Tx and Rx sides. Then the rate optimization problem forms a Markov decision problem (MDP) which can be efficiently solved by Q-learning algorithm. The simulation results indicate that the learning agent is capable of sensing the change and adjust the control policy to fit the new environment rapidly. The method is theoretically good, but considering the difficulty for implementation, the precision of channel estimation and the validity of the channel model, it is not a very practical scheme for common devices.

To better adapt the network coding system with BATS protocol, one natural attempt is to relax the bit error rate (BER) constraint to reach a higher throughput. In this paper, we first do experiments on *Minstrel* algorithm to examine whether the BER constraint relaxation brings gain to the system throughput. Then, we model the problem into a Markov Decision Problem and solve the optimal policy with Q-learning under both with-BER-constraint case and free-BER-constraint case to study the significance of such the BER constraint relaxation.

II. Methodology

Minstrel is a part of mac80211 module, thus the source codes could be found under the directory *linux\net\mac80211\rc80211_minstrel_ht.c*. I plan to read the source codes first and modify it according to my requisites. My improvement consists of two steps: (1) Make the device always transmit data with maximum-throughput-achieving rate; (2) Investigate the impact of multi-user interference and design a rate-power jointly control algorithm for multi-user case.

To validate my modifications and test the performance of the modified algorithm, some experiments should be conducted. I plan to use two industrial personal computers (*ITA-5512*) as clients to transmit data and a *Raspberry Pi 4* as the server to receive data. The industrial personal computer supports *Ath9k* driver, which uses the *Minstrel* algorithm for rate controlling. The both computers run the Ubuntu 14.04 with a 3.13 kernel. The two devices are shown in Fig. 1(a). The *Raspberry Pi 4* runs the Linux based *OpenWrt* backfire 10.03 operating system with 3.13 kernel. Thus, the *Raspberry Pi 4* could serve as an access point (AP) and I can easily start a server on it. The AP is connected via Ethernet to the campus server and associated with the client by 802.11n mode WiFi. The device is shown in Fig. 1(b). The two clients are placed at the same distance to the AP. And I use *iperf* to do UDP tests and record the throughput, delay and network jetter of both groups.



(a) clients (b) server

Fig. 1. Experiment devices

In our discussions, retransmission is not considered, thus UDP tests is more appropriate. The parameters for UDP tests is shown in Table II and the tests results are shown in the following figures.

TABLE II
UDP PARAMETERS TABLE

Parameter	Value
Bandwidth (-b)	11 Mbits/second
Duration (-t)	10 seconds
Socket buffer size (-w)	128 kbytes
packet size (-l)	32 kbytes

```
[  4] local 10.20.30.47 port 8080 connected with 10.20.30.32 port 1153
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  4]  0.0- 1.0 sec   1.3 MBytes  10.4 Mbits/sec  0.430 ms    0/   41 (0%)
[  4]  1.0- 2.0 sec   1.1 MBytes   8.5 Mbits/sec  5.996 ms    6/   40 (15%)
[  4]  2.0- 3.0 sec   1.2 MBytes   9.7 Mbits/sec  0.796 ms    1/   40 (2.5%)
[  4]  3.0- 4.0 sec   1.2 MBytes  10.0 Mbits/sec  0.403 ms    0/   40 (0%)
[  4]  4.0- 5.0 sec   1.2 MBytes  10.0 Mbits/sec  0.448 ms    0/   40 (0%)
[  4]  5.0- 6.0 sec   1.2 MBytes  10.0 Mbits/sec  0.464 ms    0/   40 (0%)
[  4]  6.0- 7.0 sec   1.2 MBytes  10.0 Mbits/sec  0.442 ms    0/   40 (0%)
[  4]  7.0- 8.0 sec   1.3 MBytes  10.4 Mbits/sec  0.342 ms    0/   41 (0%)
[  4]  8.0- 9.0 sec   1.1 MBytes   9.0 Mbits/sec  0.431 ms    0/   40 (0%)
[  4]  9.0-10.0 sec   1.3 MBytes  10.4 Mbits/sec  0.407 ms    0/   41 (0%)
[  4]  0.0-10.0 sec  12.3 MBytes   9.8 Mbits/sec  0.407 ms    7/  401 (1.7%)
```

(a) Original

```
[  3] local 10.20.30.13 port 8080 connected with 10.20.30.32 port 1313
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  2.0- 3.0 sec   1.3 MBytes   9.7 Mbits/sec  0.796 ms    1/  41 (2.5%)
[  3]  1.0- 2.0 sec   1.3 MBytes  10.4 Mbits/sec  0.221 ms    0/  41 (0%)
[  3]  2.0- 3.0 sec   1.3 MBytes  10.4 Mbits/sec  0.277 ms    0/  41 (0%)
[  3]  3.0- 4.0 sec   1.3 MBytes  10.0 Mbits/sec  1.659 ms    2/  41 (0%)
[  3]  4.0- 5.0 sec   1.3 MBytes  10.4 Mbits/sec  0.251 ms    0/  41 (0%)
[  3]  5.0- 6.0 sec   1.3 MBytes  10.4 Mbits/sec  0.215 ms    0/  41 (0%)
[  3]  6.0- 7.0 sec   1.3 MBytes   9.6 Mbits/sec  2.325 ms    3/  41 (7.3%)
[  3]  7.0- 8.0 sec   1.3 MBytes  10.4 Mbits/sec  0.254 ms    0/  41 (0%)
[  3]  8.0- 9.0 sec   1.3 MBytes   9.0 Mbits/sec  5.282 ms    5/  41 (12%)
[  3]  9.0-10.0 sec   1.3 MBytes  10.4 Mbits/sec  0.232 ms    0/  41 (0%)
[  3]  0.0-10.0 sec  13.2 MBytes  10.1 Mbits/sec  0.243 ms   11/ 410 (2.7%)
```

(b) Modified

Fig. 2. UDP test results

The figures show that the modified algorithm reaches a higher throughput of 10.1 Mbits/s with a packet loss rate of 2.7%. This simple experiment demonstrates that BER relaxation can increase throughput system throughput. To study this problem in a more theoretical way, we establish the system model and solve the best rate-power control policy.

III. PROBLEM FORMULATION

A. System Model

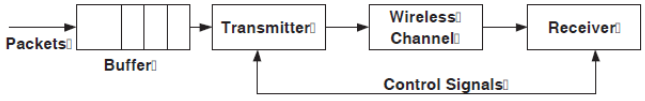


Fig. 3. System model

We consider a single-user system depicted in Fig.2. Time is divided into frames of T_f seconds. The number of packets A_i arriving to the buffer at the beginning time frame T_i follows possion distribution $p_A(a)$ with average arrival rate λ, which is given by

$$p_A(a) = \frac{\lambda^a}{a!}e^{-\lambda} \tag{4}$$

It is clear that A_i is i.i.d. so that the index i can be neglected. And all packets have the same length of L bits. The buffer has a size B and if a packet arrives when the buffer is full, it is considered as link-layer packet drop.

We consider a discrete-time block fading channel with additive white Gaussian noise (AWGN). The fading processs is a stationary and ergodic K-state Markov chain. The channel gain of the state $g, g \in$0,...,K-1, is denoted as γ_g. And the channel state is assumed to be unchanged within each time frame. The channel state transition probability is given by

$$P_G(g, g') = Pr\{G_i = g | G_{i-1} = g'\} \tag{5}$$

We denote the system state in frame i by $S_i = B(B_i, S_i)$, where B_i is the number of packets in the buffer at the beginning of frame i while G_i is the channel state. At the beginning frame i, we assume that the transmitter and the receiver have complete knowledge of S_i.

We assume that based on S_i, the transmitter can adjust its transmit power and rate. For frame i, let P_i(Watts) and U_i(packets/time frame) denote the transmit power respectively. We must have $0 <= U_i <= B_i$. In addition, we assume that $P_i \in \mathcal{P}$ where $\mathcal{P}$ denotes the set of available power levels. We use a pair (U_i, P_i) as a control action.

Let $P_b(g, u, P)$ be the function that gives the bit error rate caused by channel (BER) when the channel state is g and the transmitter rate and power are u and P respectively. $P_b(g, u, P)$ depends on the specific modulation, coding and detection schemes. We assume that a packet is in error if at least l out of its L bits are corrupted. Then we can calculate the packet error probability as

$$P_p(g, u, P) = \sum_{j=l}^{L} \binom{L}{j} P_b(g, u, P)^j (1 - P_b(g, u, P))^{(L-j)} \tag{6}$$

If we consider M-ary quadrature amplitude modulation (MQAM) and change the constellation size while keeping the symbol rate. The BER is calculated as

$$P_b(g, u, P) <= 0.2exp(-1.5\frac{P\gamma_g}{WN_0(2^u - 1)}) \tag{7}$$

where W and N_0 are bandwidth of the transmitting signal and twice the average power density of noise, u is the bits per symbol and P is the transmitting power.

B. Throughput Maximization Problem

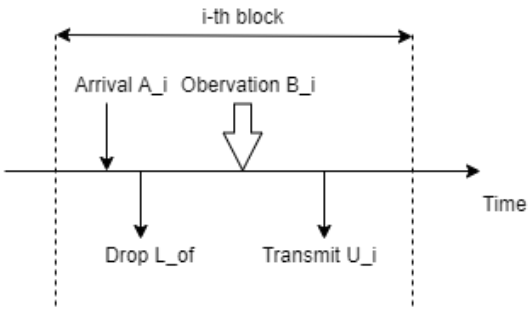


Fig. 4. Timing diagram

The system throughput is defined as the long-term average at which packets are successfully received. For an average packet arrival rate λ, a buffer overflow probability P_{of} and a packet error rate P_p, the system throughput can be calculated by

$$throughput = \lambda(1 - P_{of})(1 - P_p) \tag{8}$$

We consider the following optimization problem: At the beginning of each frame i, given the system state S_i at both transmitter and receiver, select the action (U_i, P_i) so that the system throughput is maximized subject to an average transmit power constraint $\bar{P}$.

For frame i, the process at the transmitter is demonstrated in FIg.3. Given that there are B_{i-1} packets in the buffer at the beginning of the time block and the transmission rate of the previous block is U_{i-1}, and the buffer size is *M*, the number of packets that are dropped due to buffer overflow is

$$L_{of} = max\{0, B_{i-1} + Ai - U_{i-1} - M\} \tag{9}$$

Given the transmission rate *u*, power *P*, channel state *g*, and the packet error probability $P_p(g, u, P)$, the expected number of packets lost due to transmission error is

$$L_e(g, u, P) = uP_p(g, u, p) \tag{10}$$

For fixed packet arrival rate, maximizing the system throughput is equivalent to minimizing the long-term expected packet drop due to buffer overflow and transmission error. So we have the optimization problem:

$$\min_{U_i, P_i} \quad \lim_{T\to\infty} \frac{1}{T} E\{\sum_{i=0}^{T-1} (L_{of}(B_i, U_i)) + L_e(G_i, U_i, P_i)\} \tag{11}$$

subject to:

$$\lim_{T\to\infty} \frac{1}{T} E\{\sum_{0}^{T-1} P_i\} <= \bar{P} \tag{12}$$

$$U_i \in \mathcal{U} \quad i = 0, 1, ..., T-1 \tag{13}$$

$$P_i \in \mathcal{P} \quad i = 0, 1, ..., T-1 \tag{14}$$

where the expectation means the long-term average packet drop due to buffer overflow and transmission error. And we need to solve an optimization problem for (U_i, P_i) at each time block.

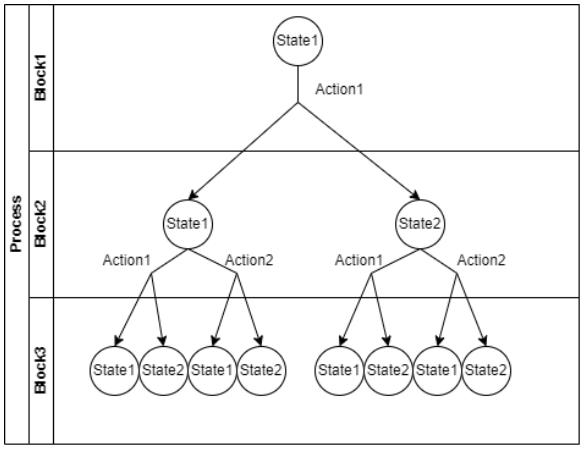


Fig. 5. Markov Decision Process example

To explain the object function (11) further, we can consider a simple case. Consider an MDP example, the state set has two states and the action set has two actions. Taking each action at a certain state gives a cost (packet loss in our case) and may cause a state transition. To get the least cost in 3 time blocks, we calculate the expected cost for each action. For example in Fig.4, to evaluate *Action1*, we calculate the weighted sum of costs for all the eight branches, which is the expectation of the costs in 3 blocks given *Action1* at *State1*.

C. Pareto Optimal Policies

Instead of directly solving the above problem, we reformulate it as a problem of minimizing a weighted sum of the long-term packet drop rate and average transmit power. In particular, we aim to minimize

$$J_{avr} = \lim_{T\to\infty} \frac{1}{T} E\{\sum_{i=0}^{T-1} C_I(B_i, U_i, G_i, P_i)\} \tag{15}$$

where $C_I(b, g, u, p)$ is the immediate cost incurred in state *(b,g)* when the control action *(u,p)* is taken.The expectation is taken with respect to C_I and the meaning for this expectation is explained in the previous section.

$$C_I(b, g, u) = P(u, g, \bar{P}_b) + \beta(L_{of}(b, u) + L_e(g, u, p)) \tag{16}$$

where β is a positive weighting factor that gives the priority to reducing packet loss over conserving power.

The problem of minimizing J_{avr} is an finite horizon average cost Markov decision process (MDP) with system state $S_i = (B_i, G_i)$, control action U_i,and immediate cost function $C_I(B_i, G_i, U_i)$. The dynamics of the system given an action is characterized by

$$P_S(s, s', u, p) = Pr\{S_{i+1} = s' | S_i = s, U_i = u, P_i = p\} \tag{17}$$

$$= Pr\{S_{i+1} = s' | S_i = s, U_i = u\} \tag{18}$$

$$= P_G(g, g') P_B(b, b', u) \tag{19}$$

because the transmission power P_i is selected to satisfy the power constraint given (G_i, U_i), and the system state S_{i+1} only depends on the channel transition and the transmission rate U_i. And the transition process of G_i and B_i is independent of each other. Futhermore,

$$P_G(g, g') = Pr\{G_{i+1} = g' | G_i = g\} \tag{20}$$

$$P_B(b, b', u) = Pr\{B_{i+1} = b' | B_i = b, U_i = u\} \tag{21}$$

$$B_{i+1} = \min\{B_i + A_{i+1} - U_i, B\} \tag{22}$$

And it is clear that B_{i+1} given (B_i, U_i) only depends on the number of packets arriving at time block T_{i+1} which is A_{i+1} following a possion distribution from our assumptions. Therefore, given the transition process of the channel and the arrival process of the packets, the transition probability matrix of the system state can be calculated.

D. Reinforcement Learning Approach

When the channel transition probability and the arrival process of the packets are perfectly known, the unconstrained MDP problem can be solved by optimal value iteration method. In this way, the optimal policy can be calculated offline.

However, the statistic information is not easy to obtain compared to the instantaneous channel gain. When only the instantaneous channel gain is available, an online algorithm is needed to learn the environment and search for the optimal solution. We use Q-learning algorithm to attain this goal. We should define the instantaneous reward for taking action (U_i, P_i) given the system state S_i, which is given by

$$r_{i+1} = -\lambda P(U_i, G_i, P_b) - (L_{of}(B_i, U_i) - L_e(G_i, U_i, P_i)) \tag{23}$$

where λ is the Lagrangian multiplier which addresses the trade-off between the cost and constraint. By properly choosing λ, the target average power constraint can be satisfied. To find the optimal control policy, the agent keeps estimating an action-value function $Q^\pi$for each state-action pair, which is the expected long-term discounted reward if the system starts at state *s*, taking action *(u,p)*, and following the same control policy π in the future,

$$Q^\pi(s, u) = E\{R_i | S_i = s, U_i = u, P_i = p\} \tag{24}$$

where π represents the control policy, and R_i is the long term discounted reward. The meaning of the expectation is the same with the expectation in equation (11).

$$R_i = \lim_{T\to\infty} \sum_{j=0}^{T} \gamma^j r_{i+j+1} \tag{25}$$

where $\gamma \in (0, 1)$ is the discounting factor. And the difference between (15) and (24) is the discounting factor γ. When $\gamma \to 1$, the minimization problem of equation (15) is equivalent to the maximization problem of equation (24).

The key idea of Q-learning is simultaneously updating the control policy and estimating the action-value function. And in the paper, we use $\epsilon - greedy$ policy, which is given by

$$a_i(S_i) = \begin{cases} a^*(S_i) & \text{with probability} \\ & 1 - \epsilon \\ a(S_i) & \text{with probability } \epsilon \end{cases}$$

where a_{S_i} is a randomly selected action in the action set, and $a^*(S_i)$ is the action that maximize that maximize the estimated action-value function,

$$a^*(S_i) = \arg \max_{b \in \mathcal{A}(S_i)} Q(S_i, b) \tag{26}$$

After taking action a_i, the system evolves to S_{i+1} and observe reward r_{i+1}. The estimation of the action-value function is updated with

$$\begin{aligned} Q(s_i, u_i) =& Q(s_i, u_i) + \alpha \times r_{i+1} + \\ & \gamma \times Q(s_{n+1}, a^*(s_{n+1}) - Q(s_i, u_i)) \end{aligned} \tag{27}$$

The Q-learning algorithm used in this project is summarized in Algorithm 1.

Algorithm 1 Multi-objective Q-learning

1: Initialization: randomly initialize *Q(s,a)* for all possible s,a, randomly choose s_0 a_0
2: T = 0
3: **repeat**
4: Take action a_i, observe r_{i+1} (23) and s_{i+1}
5: Choose a_{i+1} by (26)
6: $\bar{a} = a^*(s_{i+1})$
7: $\delta = r_{i+1} + \gamma Q(s_{i+1}, \bar{a}) - Q(s_i, a_i)$
8: $Q(s_i, a_i) = Q_(s_i, a_i) + \alpha\delta$
9: $s_i = s_{i+1}$
10: $a_i = a_{i+1}$
11: $T+ = 1$
12: **until** $T = T_{max}$
13: $policy = \arg\max_a Q(s)$

IV. SIMULATION RESULTS AND DISCUSSIONS

In this section, we present the simulation results to demonstrate the performance of the adaptive control algorithm. We compare the performance of the Q-learning algorithm with the dynamic programming approach to show the accuracy of the estimated optimal policy. And we also compare the performance of with-BER-constraint and free-BER-constraint cases to show the BER-throughput trade-off.

A. System Parameters

The packet arrival process follows Poission distribution with average rate $\lambda = 3$ Packest/Block. We assume the symbol duration is $T_b = 1$ ms and the channel has Dopller frequency $f_D = 10$ Hz. So that the correlation time satisfies $T_C \propto 1/f_D >> T_b$, which ensures a slow fading channel. The other parameters of the system are summarized in Table II. And the parameters for Q-learning are summarized in Table III.

TABLE III
SYSTEM PARAMETERS TABLE

Parameter	Value
B	15 Packets
L	100 Bits
N	100 Symbol/Block
T_b	1 ms
$N_0/2$	10^{-5} W/Hz
λ_a	3 Packets/Block
P_b	10^{-3}
α	0.2
ϵ	0.1
γ	0.95
λ_J	0.001

B. Convergence of Q-learning

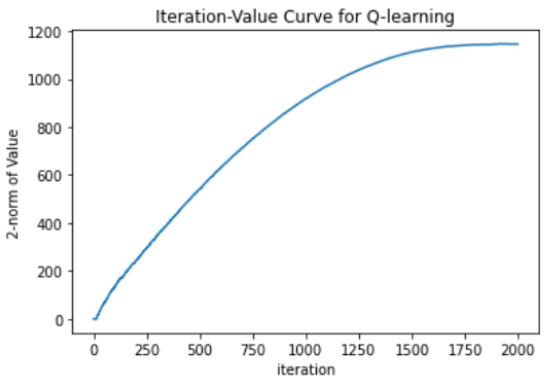


Fig. 6. Markov Decision Process example

In the simulation, there are 100 time blocks in one episode while there are 2000 episodes in total. At the beginning of each episode, the system state is randomly selected while the Q table. In this way, the Q table can iteratively estimates the value of each action at each state with equation (28). And the optimal action is the one with maximal value. From figire.5, after about 1500 iterations, the 2-norm of the action value vector converges.

C. The performance of Q-learning

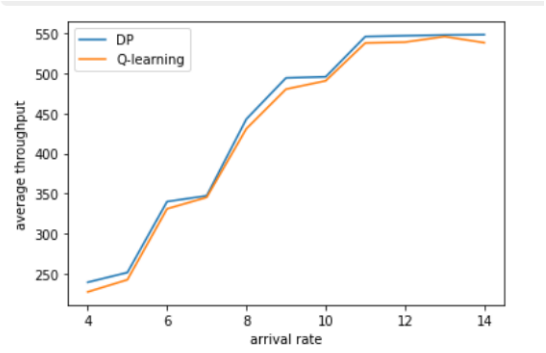


Fig. 7. Performance Comparison between DP and Q-learning

To evaluate the performance of Q-learning, we compare it with the optimal value iteration method. The figure shows the system throughput defined by equation (8) with respect to average packet arrival rate. Since the optimal value iteration method uses the system transition probability matrix, its performance is the upper bound for Q-learning estimations. The figure shows that, the Q-learning reaches a close performance with the optimal value iteration. The throughput first increases with the packet arrival rate, than achieves a maximal value when the packet arrival rate achieves a threshold. By equation (8), if the packet loss rate is constant, the throughput increases with the packet arrival rate. But when the packet arrival rate is high, the packet loss rate cannot be kept

D. The effect BER constraint

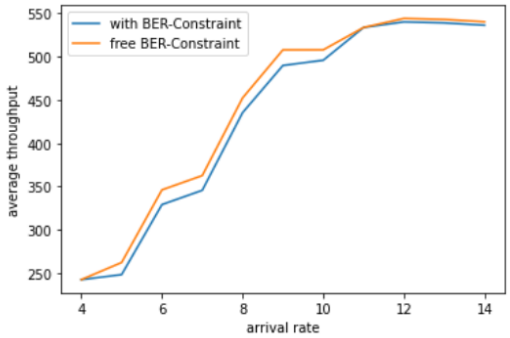


Fig. 8. Throughput Comparison

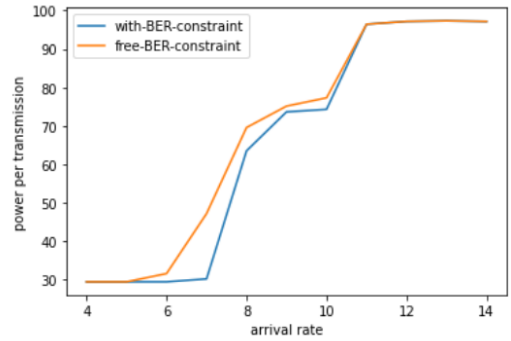


Fig. 9. Power Consumption Comparison

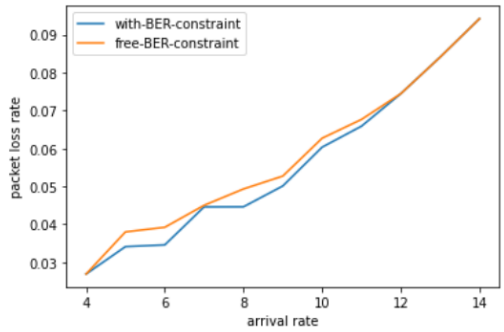


Fig. 10. Packet Loss rate Comparison

To study the effect of BER constraint relaxation, we solve the problem with optimal value iteration under with-BER-constraint case and free-BER-constraint case. Fig.8 shows the successfully transmitted packet number with respect to the average packet arrival rate. Fig.9 show the power consumption with the average average power constraint of 100 dB. Fig.10 shows the packets loss rate of the two cases. The simulation results demonstrate that BER constraint relaxation increases the system throughput with a sacrifice of sightly higher power consumption and packet loss rate. The simulation results are consistent with the experiment results in the previous section.

And we also notice that, when the packet arrival rate is very high or very low, the two schemes achieve similar throughput. When the packet arrival rate is low, the overflow rate is low, both schemes tend to use lower transmission rate to reduce power consumption or reduce the channel packet loss. When the packet arrival rate is high, both schemes tend to use higher transmission rate to avoid overflow. Thus, in the low and high arrival rate regime, the two schemes obtain similar optimal policies and get similar throughput.

V. Conclusions

In this paper, we started from the Minstrel algorithm in the Linux system and studied the link-PHY layer rate control schemes that maximize the throughput of a 1-hop communication system with finite buffer size. We modeled the problem as a Markov decision problem and used Q-learning algorithm to solve this model-free problem. Given the dynamics of the system, We used optimal value iteration to derive the optimal policy and validate the estimation results from Q-learning. We also validate that by removing the BER constraint, the system system can achieve higher throughput under the same power constraint. Meanwhile, we show the effects of packets arrival rate on system throughput.

References

[1] Huehn, T. & Sengul, C. Practical Power and Rate Control for WiFi. *2012 21st International Conference On Computer Communications And Networks (ICCCN)*. pp. 1-7 (2012)

[2] Li, X., He, H. & Yao, Y. Reinforcement learning based adaptive rate control for delay-constrained communications over fading channels. *The 2010 International Joint Conference On Neural Networks (IJCNN)*. pp. 1-7 (2010)

[3] Xia, D., Hart, J. & Fu, Q. On the performance of rate control algorithm *Minstrel*. *2012 IEEE 23rd International Symposium On Personal, Indoor And Mobile Radio Communications - (PIMRC)*. pp. 406-412 (2012)

[4] Yin, W., Hu, P., Indulska, J. & Bialkowski, K. A Method to Improve Adaptability of the Minstrel MAC Rate Control Algorithm. *Ubiquitous Intelligence And Computing*. pp. 504-518 (2010)

[5] Yin, W., Hu, P., Indulska, J. & Bialkowski, K. Performance of Mac80211 Rate Control Mechanisms. *Proceedings Of The 14th ACM International Conference On Modeling, Analysis And Simulation Of Wireless And Mobile Systems*. pp. 427-436 (2011), https://doi.org/10.1145/2068897.2068970